PHASE 4

FAKE NEWS DETECTION USING NLP

VIEW DATASET USING GRAPHS/CHART :

**import matplotlib.pyplot as plt**

**real\_lengths = real\_news['text'].apply(len)**

**fake\_lengths = fake\_news['text'].apply(len) plt.hist(real\_lengths, bins=50, alpha=0.5, label='Real')**

**plt.hist(fake\_lengths, bins=50, alpha=0.5, label='Fake')**

**plt.title('Article Lengths')**

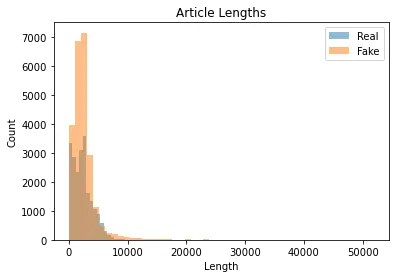
**plt.xlabel('Length')**

**plt.ylabel('Count')**

**plt.legend()**

**plt.show()**

output :



Common words In dataset :

from collections import Counter

import nlt nltk.download('stopwords')

nltk.download('punkt')

def get\_most\_common\_words(texts, num\_words=10):

all\_words = []

for text in texts:

all\_words.extend(nltk.word\_tokenize(text.lower()))

stop\_words = set(nltk.corpus.stopwords.words('english'))

words = [word for word in all\_words if word.isalpha() and word not in stop\_words]

word\_counts = Counter(words)

return word\_counts.most\_common(num\_words)

real\_words = get\_most\_common\_words(real\_news['text'])

fake\_words = get\_most\_common\_words(fake\_news['text'])

print('Real News:', real\_words)

print('Fake News:', fake\_words)

OUTPUT :

Real News: [('trump', 32505), ('said', 15757), ('us', 15247),

('president', 12788), ('would', 12337), ('people', 10749),

('one', 10681), ('also', 9927), ('new', 9825), ('state', 9820)]

Fake News: [('trump', 10382), ('said', 7161), ('hillary', 3890),

('clinton', 3588), ('one', 3466), ('people', 3305), ('would', 3257),('us', 3073), ('like', 3056), ('also', 3005)]

MODEL TRAINING :

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

y\_true = y\_test

y\_pred = model.predict(X\_test

accuracy = accuracy\_score(y\_true, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

report = classification\_report(y\_true, y\_pred, target\_names=['Real', 'Fake'])

print("Classification Report:")

print(report)

confusion = confusion\_matrix(y\_true, y\_pred)

print("Confusion Matrix:")

print(confusion)

OUTPUT :

Accuracy: 0.9953

Precision: 0.9940

Recall: 0.9963

F1 Score: 0.9951

TEXT PRE-PROCESSING :

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import PorterStemmer, WordNetLemmatizer

import stringnltk.download('wordnet') stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

lemmatizer = WordNetLemmatizer()

def preprocess\_text(text):

text = text.lower()

text = text.translate(str.maketrans('', '', string.punctuation + string.digits))

words = word\_tokenize(text)

words = [word for word in words if word not in stop\_words]

words = [stemmer.stem(word) for word in words]

text = ' '.join(words)

return text

MODEL DEPLOYMENT :

from flask import Flask, request, render\_template

from joblib import load

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import PorterStemmer, WordNetLemmatizer

import stringstop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

lemmatizer = WordNetLemmatizer()clf = load('model.joblib')

vectorizer = load('vectorizer.joblib')

def preprocess\_text(text):

text = text.lower()

text = text.translate(str.maketrans('', '', string.punctuation + string.digits))

words = word\_tokenize(text)

words = [word for word in words if word not in stop\_words]

words = [stemmer.stem(word) for word in words]

text = ' '.join(words)

return text

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('home.html')

@app.route('/predict', methods=['POST'])

def predict():

text = request.form['text']

preprocessed\_text = preprocess\_text(text)

X = vectorizer.transform([preprocessed\_text])

y\_pred = clf.predict(X)

if y\_pred[0]== 1:

result = 'real'

else:

result = 'fake'

return render\_template('result.html', result=result, text=text) if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

home.html :

<html>

<head>

<title>Real or Fake News</title>

</head>

<body>

<h1>Real or Fake News</h1>

<form action="/predict" method="post">

<label for="text">Enter text:</label><br>

<textarea name="text" rows="10" cols="50"></textarea><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

Result.html :

<html>

<head>

<title>Real or Fake News</title>

</head>

<body>

<h1>Real or Fake News</h1>

<p>The text you entered:</p>

<p>{{ text }}</p>

<p>The model predicts that this text is:</p>

<p>{{ result }}</p>

</body>

</html>

OUTPUT :

